TypeScript Meetup v0.2

10 min	Around the room introductions
10 min	Opening comments - Anders Hejlsberg
20 min	TypeScript in the trenches - Eric Gamma
20 min	TypeScript update - Luke Hoban
20 min	TypeScript Q&A session - Luke, Anders, Steve Lucco, and Team
20 min	**Break: Eat, drink, and Socialize**
10 min	Bing and TypeScript - Kiran Badam & Sarvesh Nagpal
10 min	Cloud Make: Using TypeScript for Builds and WorkFlows - Wolfram Schulte
10 min	TypeScript language experiments - Ron Buckton
10 min	Xbox Music and TypeScript - Dan Rodgers (not recorded/streamed)
10 min	Using TypeScript to simplify JavaScript hosting - Paul Vick (not recorded/streamed)

Contact: stevenic

Thank You!

TypeScript Momentum

Community

Over 3000 CodePlex posts, 100 forks, 1000 StackOverflow questions, and 300 feature requests

Ecosystem

Over 180 .d.ts library definitions covering more than 90% of popular JavaScript frameworks.

Tool Support

IDEs: WebStorm, WebMatrix, JSBin, Web Essentials, Sublime Text, vi, Emacs, FlashDevelop, Brackets
Build: Heroku, Ruby, grunt, ASP.NET, node, compile-in-client
Testing: Chutzpah, tsUnit
Preprocessing: Bakehouse
Typings management: tsd

TypeScript 0.9

Generics

Generic classes, interfaces, and methods: The top requested feature by users

Overloading on Constants

Examples include createElement, getElementsByTagName, addEventListener

Other Features

Enum types, 'export =', function/module and class/module merging, methods in object literals

Compiler

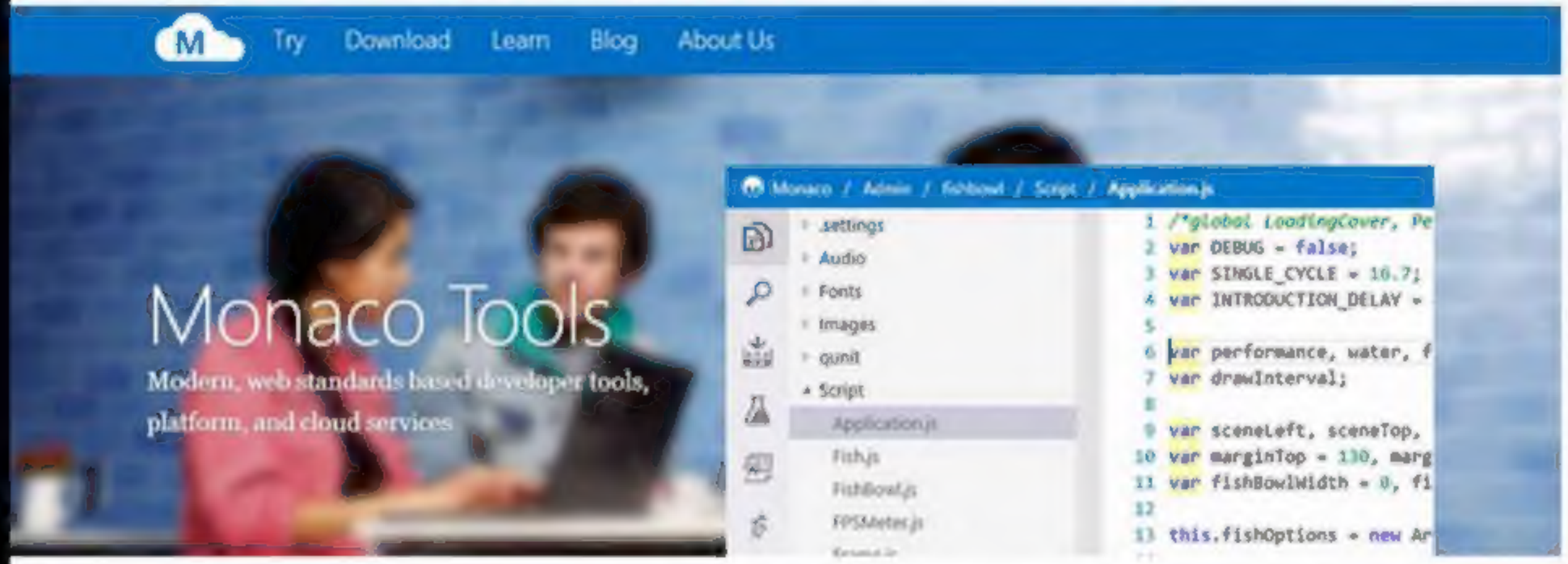
Incremental parser, pull-model type checker, scaling to 100K+ line projects

Application Scale JavaScript

TypeScript in the Trenches

Erich Gamma
Microsoft

http://monacotools



Easily get started

No long waits for installation and configuration. Just log in and get coding!

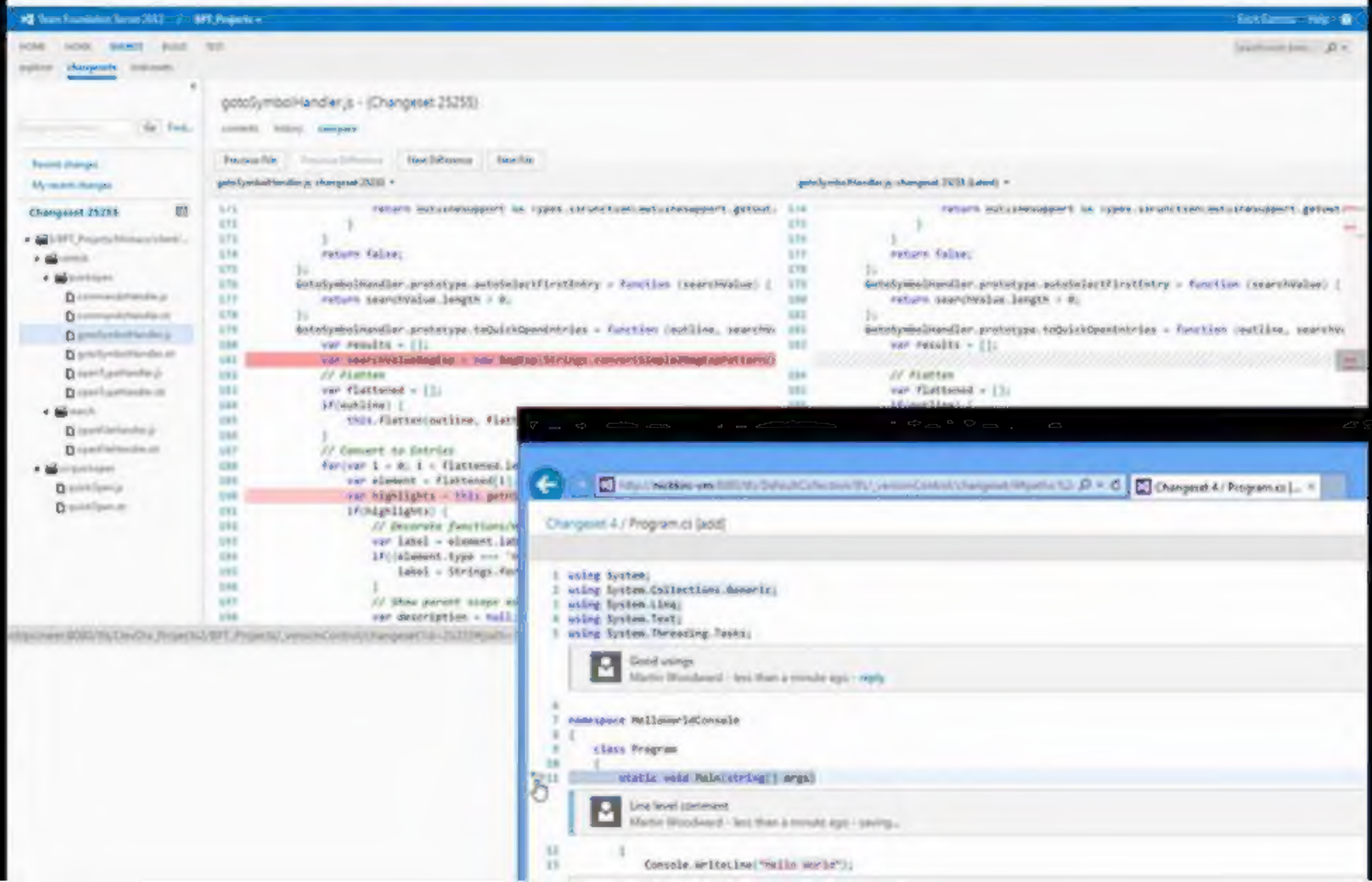
Code in the cloud with confidence

Develop online with rich, contextual features like IntelliSense, Source Assist, Quick Open and many more!

Platform of reusable components

We've built Monaco as a set of simple, yet extensible modules that you can adopt and tweak yourself, based on your requirements and scenarios

Team Foundation Server



dafny

Microsoft Research

Is this program correct? Ask dafny!

```
1  function Ackermann(m: int, n: int): int
2    // The following lexicographic pair allows Dafny to prove termination.
3    // Still, you may not want to sit around and wait for a call to Ackermann
4    // to terminate.
5    decreases m, n;
6  {
7    if m <= 0 then
8      n + 1
9    else if n <= 0 then
10     Ackermann(m - 1, 1)
11   else
12     Ackermann(m - 1, Ackermann(m, n - 1))
13 }
14
```

ask dafny

home | tutorial | video | permalink

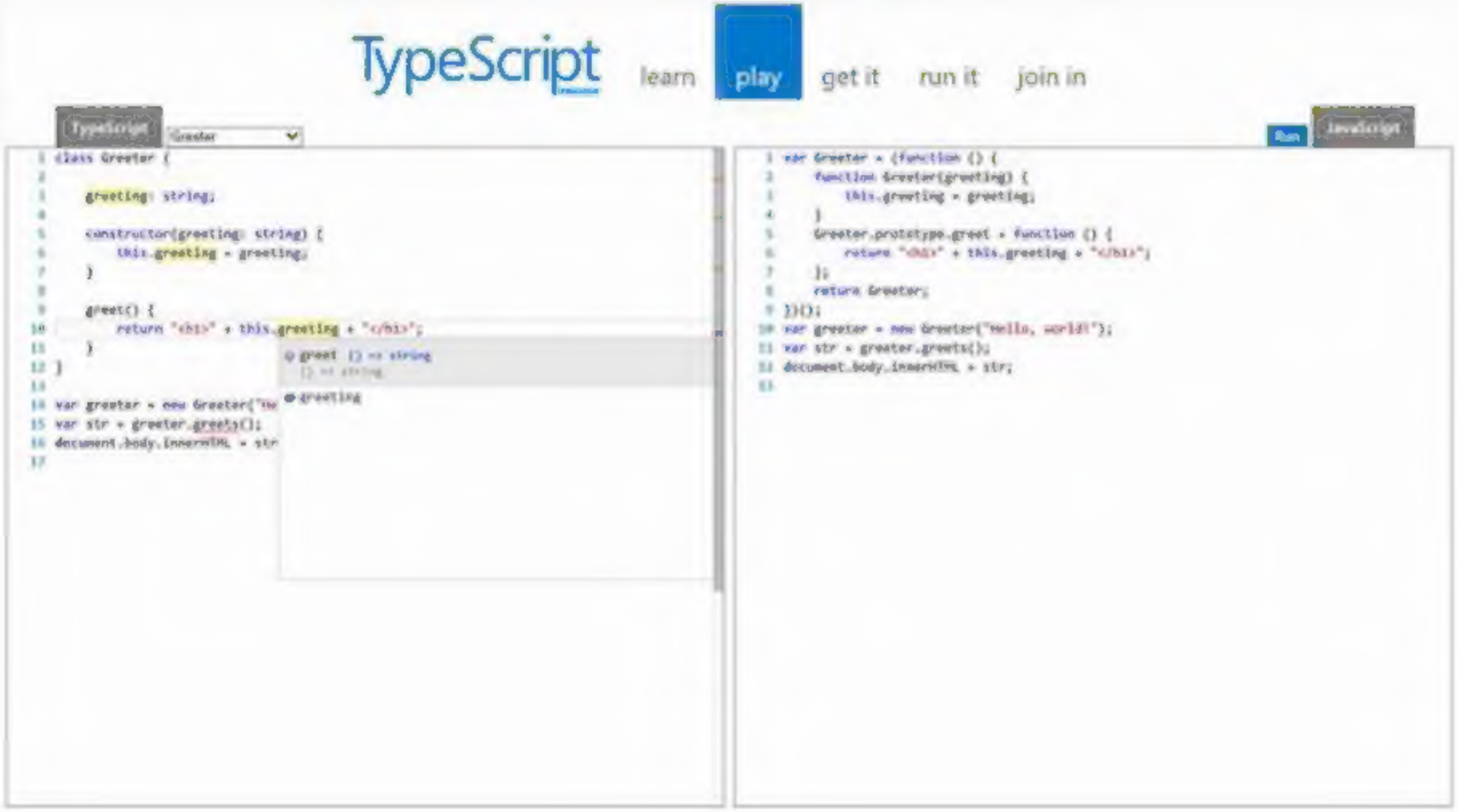
Dafny program verifier finished with 1 verified, 0 errors

samples
Hello
Fibonacci
CountToN

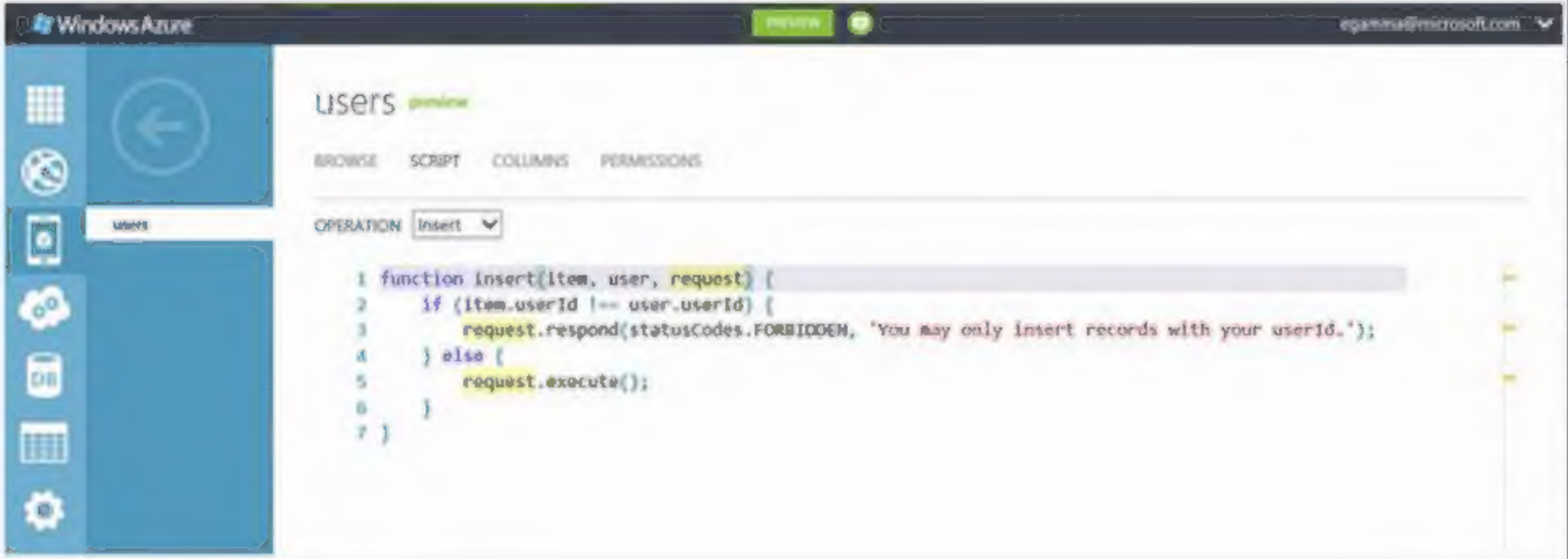
about Dafny - A language and program verifier for functional correctness

Dafny is an imperative, object-oriented programming language with classes and inductive datatypes, and specification constructs for describing intended behavior. The Dafny verifier checks that programs live up to their specifications.

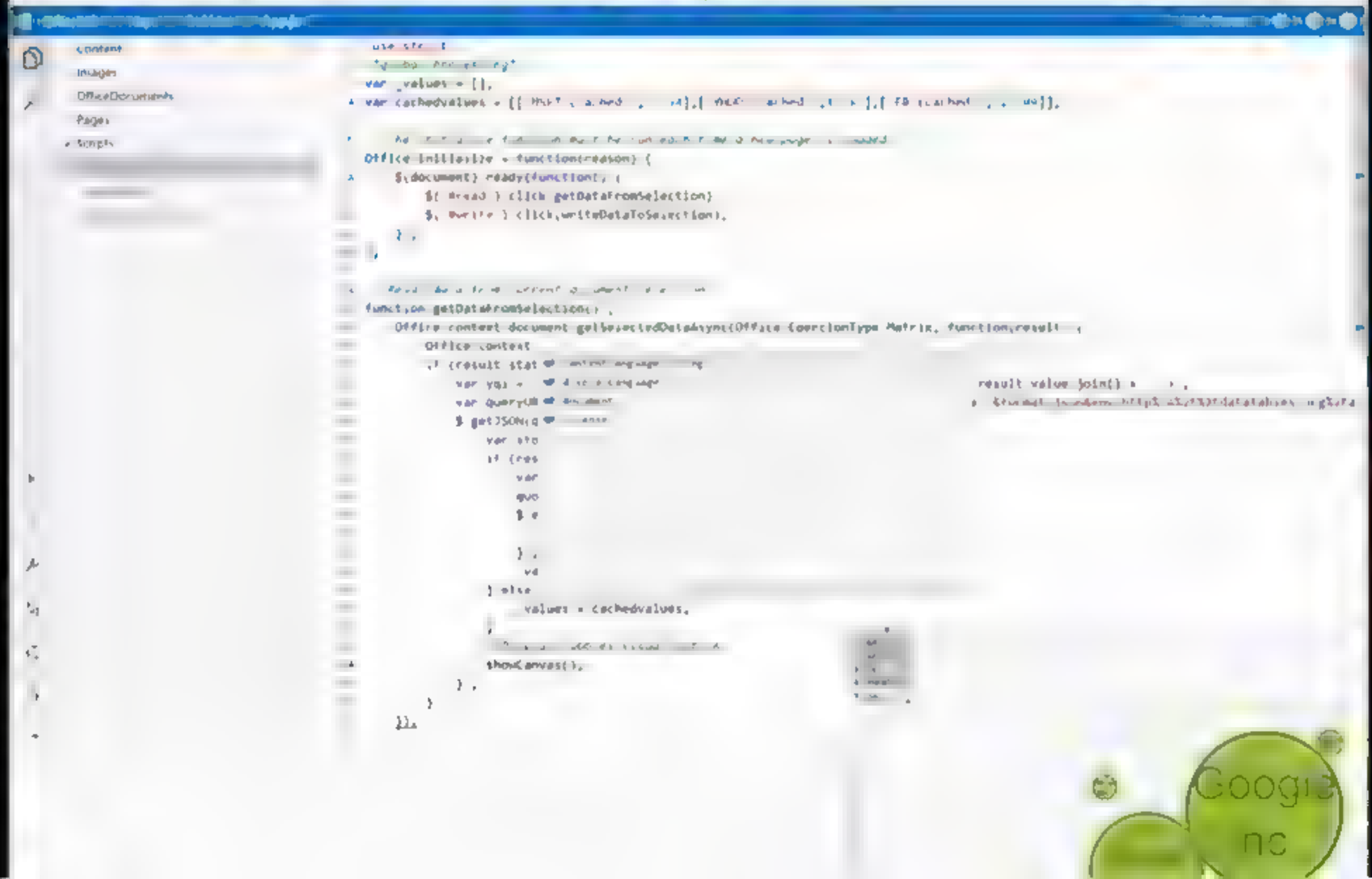
TypeScript Playground



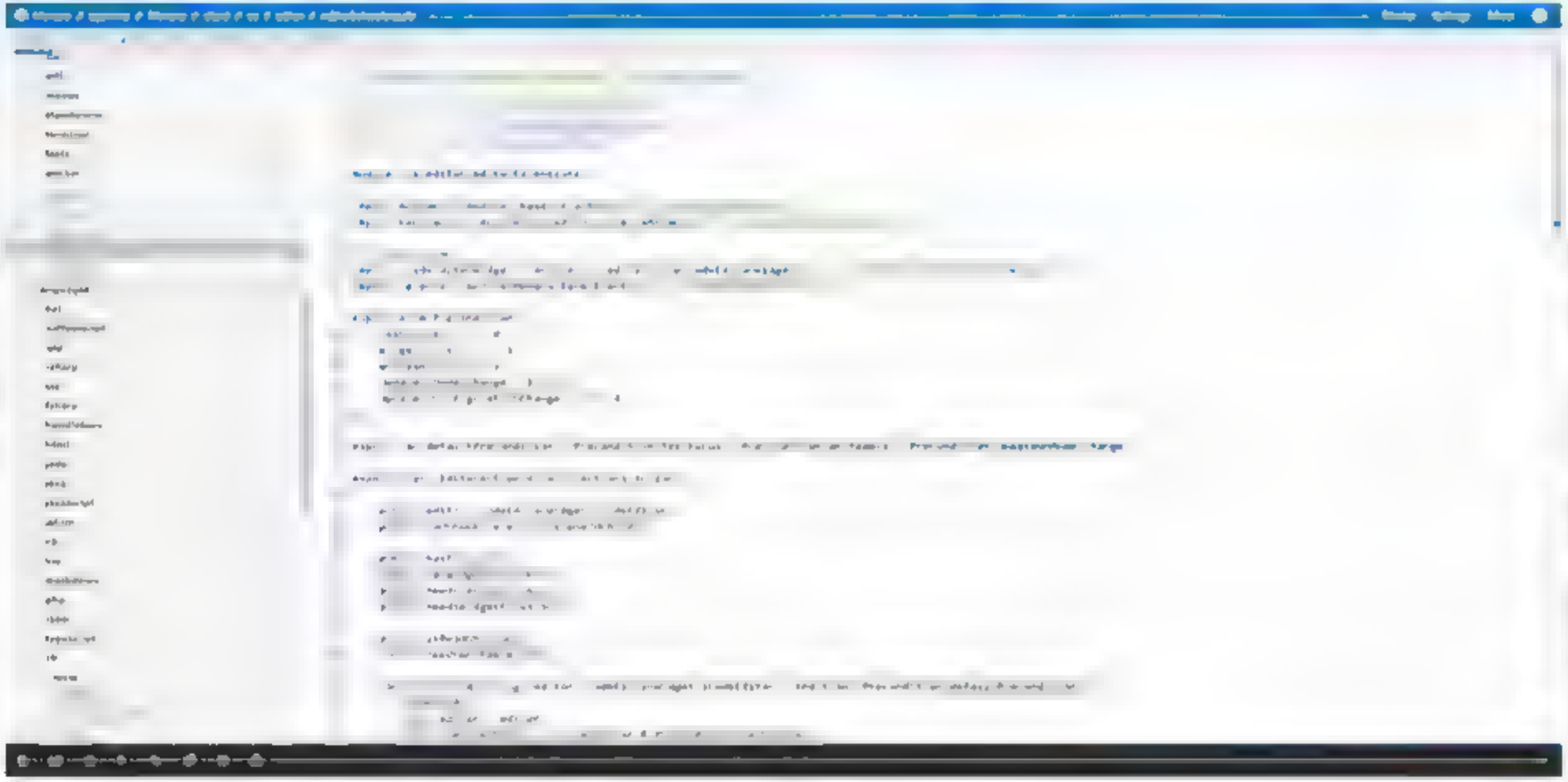
Azure


Windows Azure
preview
egammal@microsoft.com
users
users preview
BROWSE
SCRIPT
COLUMNS
PERMISSIONS
OPERATION
Insert
function insert(item, user, request) {
if (item.userId !== user.userId) {
request.respond(statusCodes.FORBIDDEN, 'You may only insert records with your userId.');
} else {
request.execute();
}
}

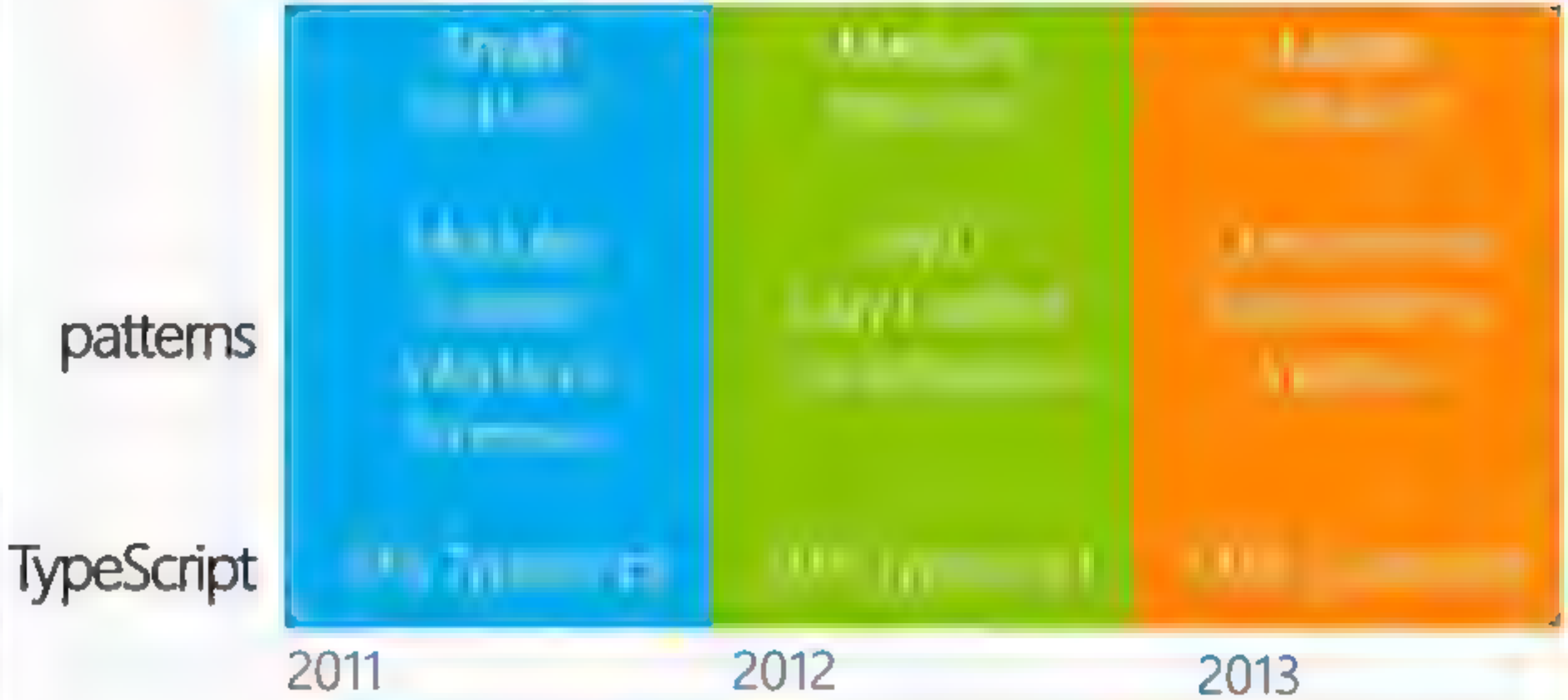
Office 365 – Napa



Monaco @ Work - Dog Fooding



Our Journey



Inside

```
interface IRenderer {
    getHeight(tree: ITree, element: any): number;
    render(tree: ITree, element: any): void;
}
```

Interfaces at Work

Options JSON

```
interface IOptions {
    hideButtons?:bool;
    okOnFocusLost?:bool;
}
```

Interfaces at Work

```
interface IMeasurementCallback {
    (accessor: IMeasurementAccessor):void;
}
```

Interfaces at Work

```
interface IEventMap {
    [name: string]: EventListener[];
}
```

Interfaces at Work

External Types

```
interface JQuery {
  addClass(classNames: string): JQuery;
  addClass(func:(index: any, currentClass: any)=>JQuery);
  attr(attributeName: string): string;
  attr(attributeName: string, value: any): JQuery;
  ....
}
```

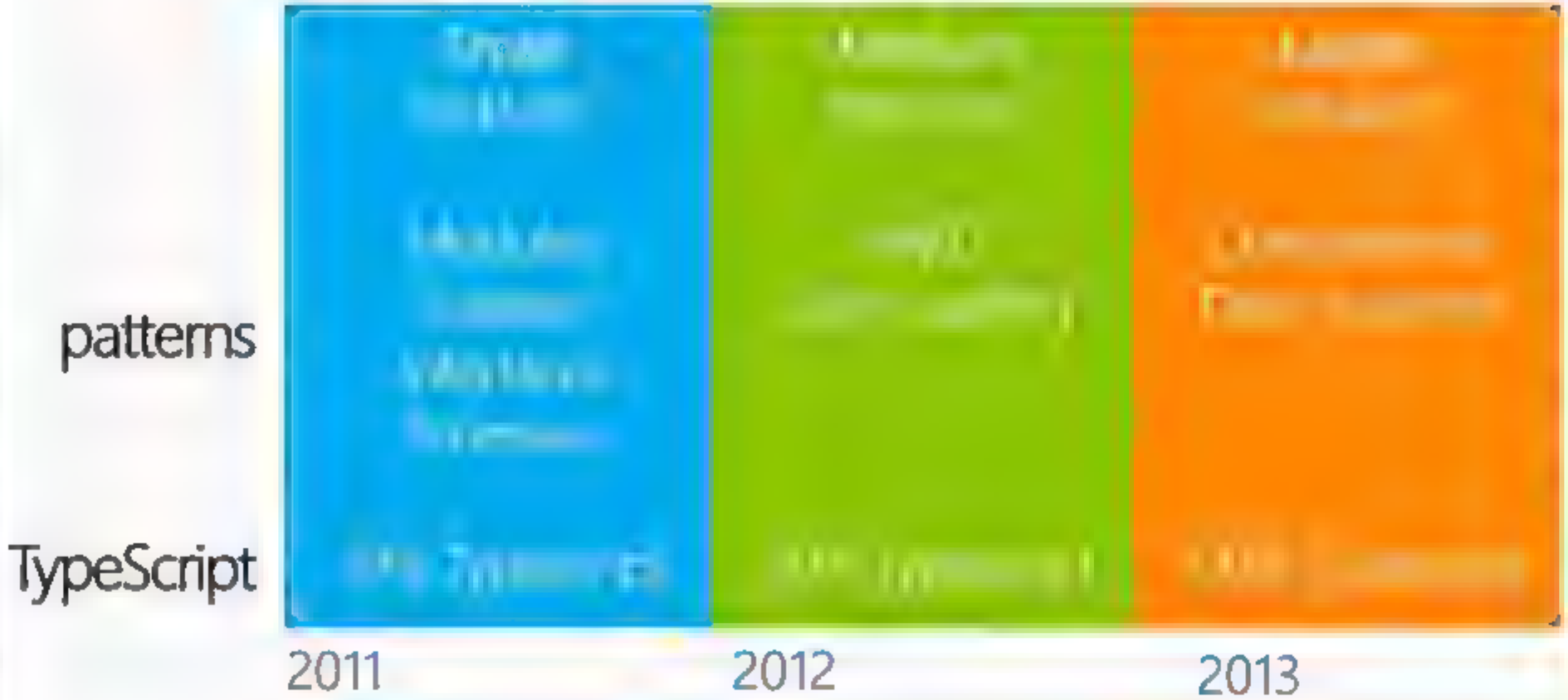
Demo

Type definitions

TypeScript Type Definitions

https://github.com/borisyankov/DefinitelyTyped

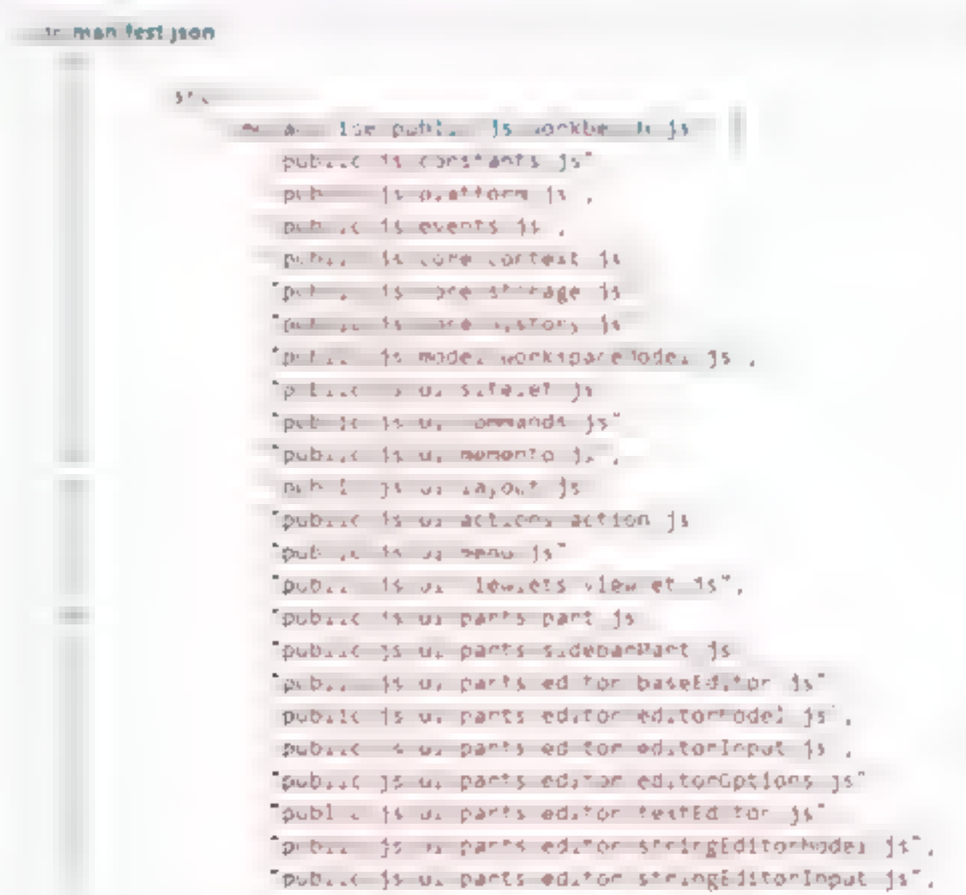
Our Journey



Growing Pains

Managing scripts and their order

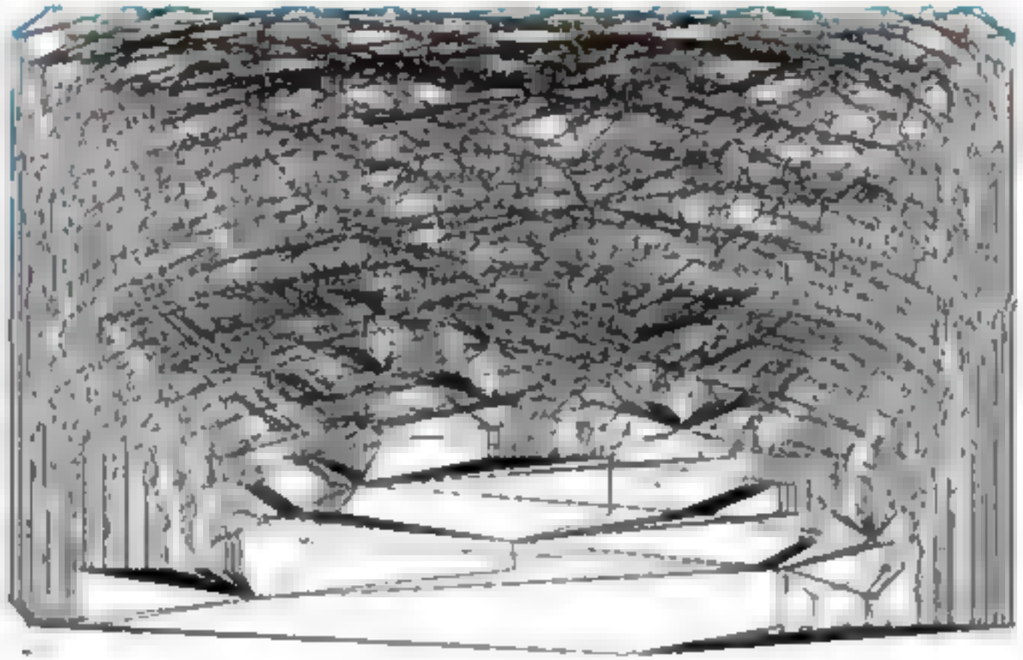
Manually edited lists of scripts



Growing Pains: Dependencies...

It is easy to import an internal module...

It is easy to contribute to a module...



"our dependency graph was such a mess that each area had a dependency on just about every other area." –Nick

AMD to the Rescue

```
define(id?, dependencies?, factory)
```

```
define([
    'vs/base/lib/winjs',  vs/editor/zoneWidget'],
    function(WinJS, ZoneWidget) { ... }
);
```

TypeScript: External Modules

TypeScript supports code generation for different module systems

```
import widget= import('vs/base/widget');

import http= import('http');
```

```
tsc --module amd app.ts
```

```
define([…], function(…) {…}
```

```
tsc --module commonjs HttpServer.ts
```

```
var http= require("http");
```

Before/After

AMD in TypeScript

```
define(['.../winjs.base', '.../zoneWidget'],
    function(WinJS, ZoneWidget) { ... }
);
```

TypeScript

```
import WinJS= import( vs/base/lib/winjs );
import ZoneWidget = import( vs/editor/zoneWidget');
```

AMD Applied – Self Contained Modules

Support à la carte consumption

Expressing CSS dependencies

CSS AMD loader plugin

TypeScript pragma

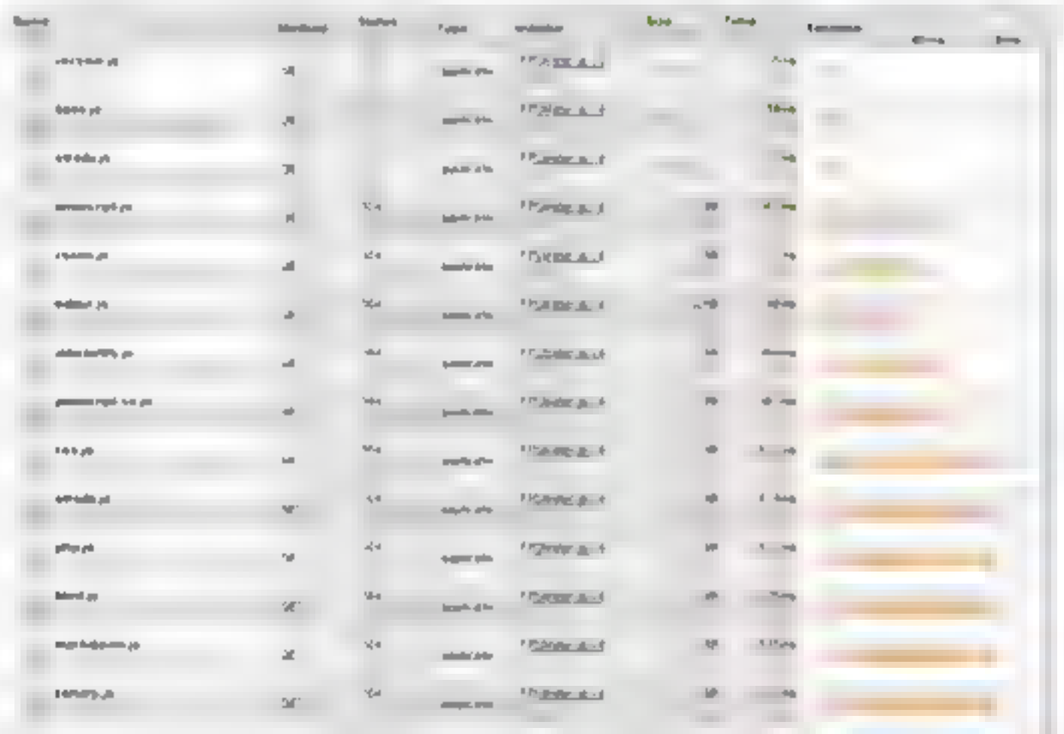
```
/// <amd-dependency path="vs/css!./actionbar" />
```

Growing Pains

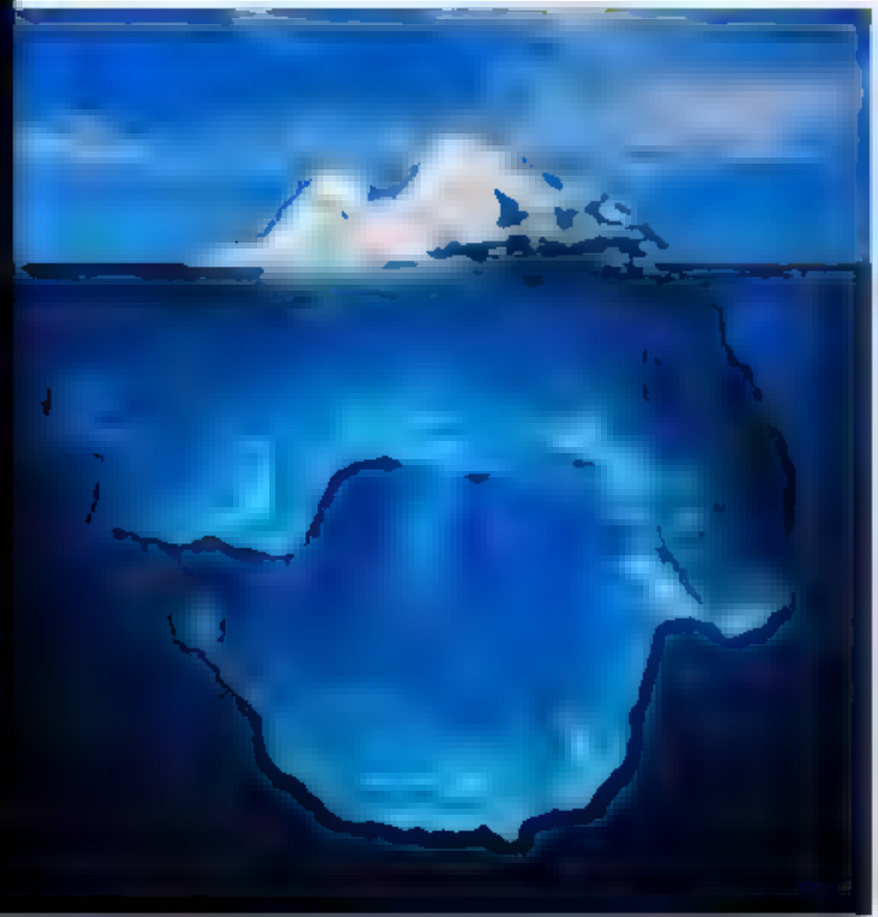
Start up time

Eager loading of scripts and CSS

Slow startup: #requests, data transfer



Lazy Loading Contributions



csharp.contribution.ts

```
modeRegistry.registerMode(
    ['text/x-csharp'],
    new Platform.Descriptor(
        'vs/languages/csharp/csharp',
        'CSMode')
);
```

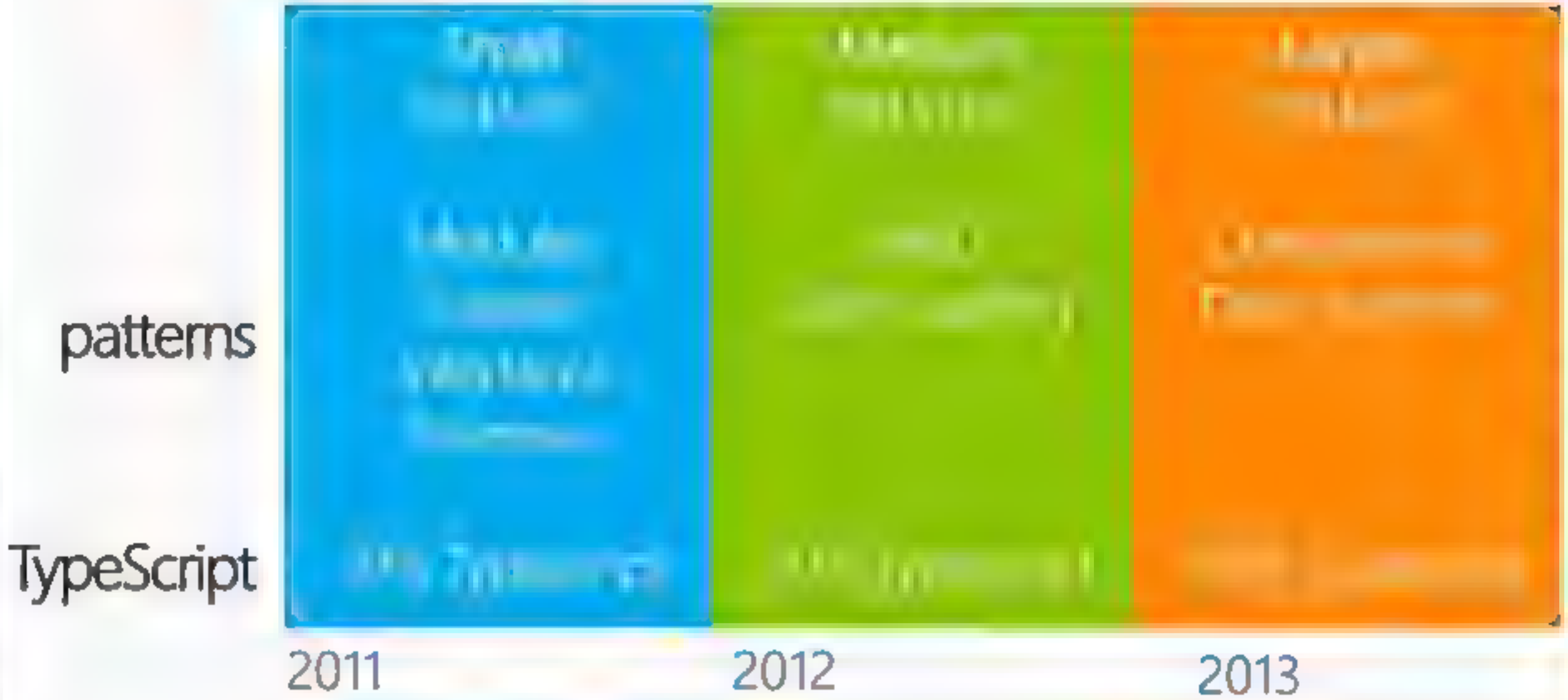
csharp.ts

```
export class CSMode extends
modesExtensions.AbstractMode {
    constructor() {
        super('vs.languages.csharp');
    }
   // lots of code ....
}
```

After the AMD Migration Impressions

"It feels like [illegible] showered. Self contained modules, no more cycles, no more globals, clean file system structure" -Alex

Our Journey



100% TypeScript

"Writing JavaScript code in a large project is like carving code in stone" –Alex

Migration is code clean-up and real work

Velocity around 300 LOCs per hour

"As I did conversions, I began typing various object literals I was passing around as interfaces. Soon enough, I realized how inconsistent I was, the same data was flowing around in at least 3 different formats. This is because of the easiness through which you can create literals in JavaScript .... Need some placeholder for data?... Just create a new literal object." --Alex

Even More Growing Pains

"Writing JavaScript code in a large project is like carving code in stone" –Alex

Tooling slow down, translation time

Tools need to digest lots of source to infer types

Hard coded dependencies on context

Decreased testability

Difficulty to reuse components in different contexts

Componentization

Tooling slow down, translation time

Tools need to digest lots of source to infer types

Reuse TypeScript code as 'binary' JS components with a declarations file

Example using TypeScript language services as a component

Compiler and language services > 30kLOC of TypeScript

```
tsc --declarations --out typescriptservices.js typescript.ts
```

Dependency Injection

Dependencies on *services* provided by container

Decreased testability

Need for configuration flexibility

Services (20)

Selection, Progress, Logging, History, Events, Configuration, Telemetry, Requests, ...

Dependency Injection

```
export interface IProgressService {
    show(total?:number, delay?:number):IProgressRunner;
}

export interface IProgressServiceConsumer {
    injectProgressService(service:IProgressService):void;
}

export class Explorer implements Services.IProgressServiceConsumer{
    private progressService:Services.IProgressService;
    public injectProgressService(service:Services.IProgressService) {
        this.progressService = service;
    }
    //....
}
```

TS Retrospective

We were on the bleeding edge...

...but we expected it and had plenty of band aid

We would do it again, the benefits outweigh the pains

Code readability, refactoring agility, tooling, fun

We would start with TypeScript (and AMD) from the beginning

http://www.typescriptlang.org/